Réécriture et détection d'implication textuelle

Paul Bedaride, Claire Gardent

TALN 2008, Avignon, 9–13 juin 2008

Réécriture et Détection d'Implication Textuelle

Paul Bedaride [1] Claire Gardent [2]
(1) Université Henri Poincaré/LORIA, Nancy
(2) CNRS/LORIA, Nancy
Paul.Bedaride@loria.fr,Claire.Gardent@loria.fr

Résumé. Nous présentons un système de normalisation de la variation syntaxique qui permet de mieux reconnaître la relation d'implication textuelle entre deux phrases. Le système est évalué sur une suite de tests comportant 2 520 paires test et les résultats montrent un gain en précision par rapport à un système de base variant entre 29.8% et 78.5% suivant la complexité des cas considérés.

Abstract. We present a system for dealing with syntactic variation which significantly improves the treatment of textual implication. The system is evaluated on a testsuite of 2 520 sentence pairs and the results show an improvment in precision over the baseline which varies between 29.8% and 78.5% depending on the complexity of the cases being considered.

Mots-clés : Normalisation syntaxique, Détection d'implication textuelle, Réécriture de graphe.

Keywords: Syntactic normalisation, Recognising Textual Entailment, Graph rewriting.

1 Introduction

La reconnaissance d'implications textuelles consiste à déterminer si un texte en implique un autre. Pour le traitement sémantique des textes, c'est une tâche fondamentale qui permet, par exemple, d'aider à déterminer si une phrase est une réponse à une question (la réponse implique la clôture existentielle de la question) ou de détecter les phrases redondantes dans un résumé (une phrase implique l'autre). Plus généralement, le traitement de l'implication textuelle permet de détecter des relations d'implication, d'équivalence et de contradiction entre fragments textuels et donc de raisonner sur le sens des textes.

Au cours des trois dernières années, l'évolution de la campagne RTE (Recognising Textual Entailment (Dagan *et al.*, 2005)) a clairement montré que la reconnaissance d'implications textuelles passe par un traitement en profondeur des données textuelles. En effet, les résultats obtenus montrent que les approches basées sur les sacs de mots donnent des résultats décevants (précision d'environ 60% avec une « baseline » de 50%), et qu'il est nécessaire à la fois de pouvoir faire abstraction des différences syntaxiques qui peuvent distinguer des textes sémantiquement équivalents et de pouvoir raisonner sur le sens des textes.

Dans cet article, nous nous concentrons sur la première de ces problématiques et présentons un système pour la reconnaissance d'implications textuelles dédié au traitement de la variabilité syntaxique de l'anglais entre textes à même structure prédicative (voir la Figure 1 pour des exemples d'implication textuelle).

(1)	*« John hates the man that hit Mary. »*	$\Rightarrow_T$	*« Mary was hit by the man that is hated by John. »*
(2)		$\Rightarrow_T$	*« John hates a man. »*
(3)		$\not\Rightarrow_T$	*« John hates Mary. »*

FIG. 1 – Examples d'implications texutelles

Nous commençons (section 2) par présenter une méthode permettant de coupler le Stanford Parser avec une phase de normalisation des sorties et de reconnaissance d'implication textuelle. En bref, l'approche proposée consiste à « normaliser » les graphes de dépendance sémantiquement équivalents par application de règles de réécriture ; à traduire les représentations normalisées en formules de la logique du premier ordre ; et à tester l'implication par le biais d'un raisonneur.

Nous illustrons ensuite l'impact de la phase de normalisation sur la reconnaissance d'implications textuelles par une évaluation basée sur une suite de tests faisant intervenir des variations syntaxiques (section 3). Dans cette suite, chaque élément est une paire de textes $\langle T_1, T_2 \rangle$ annotée comme étant ou non un cas d'implication textuelle. En outre, lorsque T_1 implique textuellement T_2, T_1 et T_2 ne diffèrent que par la syntaxe (e.g., variation actif/passif). L'évaluation montre que, sur cette suite de tests, la phase de normalisation intégrée par notre approche permet d'améliorer la précision du système de 29.8% pour les cas simples (un seul verbe conjugué dans la phrase impliquée T_2) et de 78.5% pour les cas plus complexes (deux verbes conjugués dans T_2).

Nous concluons (section 5) en indiquant des pistes à poursuivre d'une part pour généraliser l'approche à l'ensemble des variations syntaxiques, et d'autre part pour l'enrichir avec des connaissances autres que syntaxiques (sémantique lexicale, savoir encyclopédique, etc.).

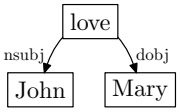

FIG. 2 – « John loves Mary »

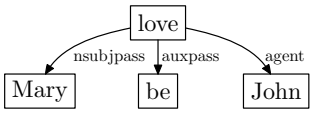

FIG. 3 – « Mary is loved by John »

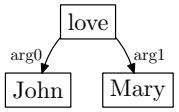

FIG. 4 – Représentation normalisée

2 Approche Proposée

L'approche présentée dans cet article vise à associer à un ensemble de structures syntaxiquement distinctes mais sémantiquement équivalentes, une représentation sémantique « normalisée » unique. Nous utilisons les familles de la grammaire XTAG pour cataloguer ces ensemble. En effet XTAG est découpée en familles regoupant chacune les différentes variations syntaxique pour une classe de verbe. Par exemple la famille Tnx0Vnx1 représentant les verbes transitifs, contient les variations nx0Vnx1 et nx1Vbynx0 représentants respectivement l'actif et le passif. Les graphes de dépendances des Figures 2 et 3 représentent ces variations, et notre objectif est de les transformer en une représentation sémantique unique correpondant à la Figure 4.

En effet, si elles présentent généralement des différences de type fonctionnel ou pragmatique (différentes structures communicatives par exemple), les variantes telles que la variante actif/passif véhiculent un contenu informationnel identique et doivent donc pouvoir être reconnues comme sémantiquement équivalentes par un reconnaisseur d'implications textuelles. Nous proposons de reconstruire ces équivalences à partir des graphes de dépendances produits par l'analyseur de Stanford modifiés par un système de réécriture. Par exemple, pour normaliser les structures actif/passif données en Figure 2 et 3, nous définissons des règles de réécriture qui peuvent être représentées graphiquement de la façon suivante :

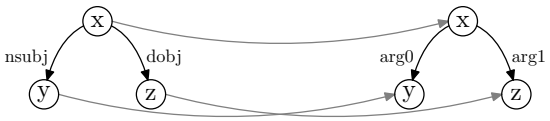

FIG. 5 – Régle de réécriture de l'actif

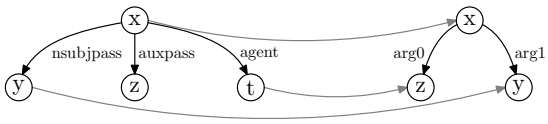

FIG. 6 – Régle de réécriture du passif

En outre, pour détecter les relations d'implication textuelle entre deux textes T_1 et T_2, les représentations sémantiques créées par les règles de réécriture sont traduites en formules de la logique du premier ordre et l'implication entre deux textes testée par inférence logique : un texte T_1 implique un texte T_2 si et seulement si l'une des représentations sémantiques normalisées de T_1 implique l'une des représentations sémantiques normalisées de T_2.

En résumé, l'approche proposée comprend les étapes suivantes :
- L'analyse syntaxique est faite par le Stanford Parser (SP)
- Des règles de réécriture sont définies à partir de la grammaire XTAG pour traduire les graphes de dépendances produits par le SP en représentations sémantiques normalisées
- Pour chaque phrase analysée, ces règles de réécriture sont appliquées aux cinq premières sorties du Stanford Parser
- Les représentations sémantiques produites par la réécriture sont traduites en formules de la logique du premier ordre
- Étant donnés deux textes T_1 et T_2, T_1 implique T_2 ssi (la traduction en logique du premier ordre de) l'une des représentations sémantiques normalisées de T_1 implique (la traduction en logique du premier ordre de) l'une des représentations sémantiques normalisées de T_2.

Dans le reste de cette section, nous allons tout d'abord présenter les outils que nous utilisons, puis nous expliciterons les principaux points de notre approche. Nous présentons ensuite (section 3) les résultats d'une évaluation basée sur une suite d'exemples faisant intervenir la variation syntaxique.

2.1 Analyse syntaxique et réécriture

Le Stanford Paser et ses graphes de dépendance Le Stanford Parser (Klein & Manning, 2003) est un analyseur syntaxique possédant une bonne précision. Il est entrainé sur ***la section ? ? ? du *** le Penn Treebank et obtient un score F de 83,36% sur la section 23 de celui-ci. Afin d'obtenir une sortie plus sémantique, il a été étendu avec un module permettant d'extraire une structurede dépendance typés à partir d'un arbre syntaxique (de Marneffe *et al.*, 2006). La sortie finale est un graphe de dépendances dont les noeuds sont étiquetés avec les mots de la phrase analysée et dont les arcs sont orientés et étiquetés avec des relations grammaticales tels que objet, sujet, ou modifieur. Au total, il existe 48 relations grammaticales différentes.

La réécriture La réécriture est une technique permettant de modéliser les concepts de réduction et de simplification. Par exemple, la règle de réécriture $r_1 : x * y + x * z \rightarrow x * (y + z)$ permet de factoriser le calcul $5 * 6 + 5 * 7 + 5 * 8$ en $5 * ((6 + 7) + 8)$.

Un système de réécriture est un ensemble de règles de réécriture de la forme $l \rightarrow r$. Une telle règle s'applique à un objet t si celui-ci contient une instance du membre gauche l, c'est-à-dire un sous-objet que l'on peut identifier à l, cette étape s'appelle le filtrage. L'objet t se réécrit alors en un nouvel objet t', obtenu en remplaçant l'instance de l par l'instance du membre droit r correspondante.

Typiquement, l'application d'un système est régi par une stratégie permettant de définir l'ordre d'application des règles, ainsi que la façon dont une règle est appliquée. Ainsi suivant la façon d'appliquer la règle ci-dessus (de gauche à droite ou de droite à gauche), le résultat obtenu pourra être $5 * ((6 + 7) + 8)$ ou $5 * (6 + (7 + 8))$.

Utilisés dans des domaines très variés (calcul formel, algèbre combinatoire, sémantique opérationnelle, etc.), la réécriture a été étudiée pour différents types d'objets syntaxiques (e.g., mots, termes, lambda termes) et en particulier, pour les graphes. Pour normaliser les graphes de dépendances produits par le Stanford Parser, nous utilisons le système de réécriture de graphes GrGen.Net(Kroll & Geiß, 2007). Dans les paragraphes suivant, nous allons expliquer comment ce système permet de définir des graphes, des régles de réécriture ainsi que des stratégies de réécriture.

Graphes GrGen.Net. GrGen.Net permet de travailler sur des graphes dont les arcs sont orientés et typés et dont les noeuds sont typés et peuvent être étiquetés avec des attributs typés. Les différents types peuvent en outre être structurés dans une hiérarchie d'héritage. GrGen.Net a donc toute l'expressivité requise pour réécrire les graphes de dépendances produits par le Stanford Parser. Dans GrGen.Net, on peut donc représenter les graphes de dépendances de la façon suivante. Les mots étiquetant les noeuds des graphes de dépendances seront représentés par des attributs sur les noeuds et les arcs étiquetés de relations grammaticales des graphes de dépendances par des arcs typés.

Règles de réécriture GrGen.Net. Une règle de réécriture de la forme $l \rightarrow r$ définit un motif de filtrage l et une réécriture r de la structure identifiée par le motif. Dans GrGen.Net, un motif de filtrage est un graphe sous-spécifié où les contraintes exprimées peuvent contraindre le type des noeuds et des arcs et les attributs des noeuds. Il est également possible d'exprimer des contraintes négatives (par exemple, pour exiger qu'un noeud n'ait pas plus de trois arcs sortants). Les opérations de réécriture permises sont la duplication, la suppression ou l'ajout d'information. Elles sont exprimées par la partie droite de la règle qui spécifie comment réécrire la structure identifiée par le motif de filtrage.

Stratégies de réécriture GrGen.Net. Pour déterminer l'ordre d'application des règles, GrGen.Net permet de regrouper les règles en séquences puis d'imposer différents types de contraintes sur l'exécution de ces séquences. Une séquence de règles peut être disjonctive, conjonctive ou itérative. Deux ou plusieurs séquences de règles peuvent être combinées par concaténation (la combinaison réussit si au moins l'une des séquences de règles peut être appliquée avec succès), par un AND transactionnel (pour que la combinaison de séquences soit exécutée, il faut que chacune des séquences spécifiées puisse être appliquée dans l'ordre spécifié) et par un XOR ou disjonction exclusive (la combinaison de séquences réussit dès que l'une des séquences spécifiées peut être appliquée). Il est par ailleurs possible d'imposer soit une application « globale » (la règle est appliquée à toutes les structures filtrées par son motif dans le graphe réécrit) soit une application « unique » (la règle n'est appliquée qu'à une seule structure).

2.2 Définition et application des règles de réécriture

Les règles de réécriture nécessaires à la normalisation des structures de dépendances produites par le Stanford Parser sont définies manuellement. Afin de prendre en compte la majeur partie des variations syntaxiques de l'anglais, nous avons décider de baser notre approche sur le grammaire XTAG. Cette grammaire a l'avantage d'avoir une bonne couverture de l'anglais, ainsi que de regrouper les variations syntaxiques en familles. Les règles de réécriture nécessaires à la normalisation des structures de dépendances produites par le Stanford Parser sont définies manuellement. À partir de la grammaire XTAG de l'anglais (Group, 2001), nous commençons par établir un ensemble de phrases illustrant, pour un cadre de sous-catégorisation donné, l'ensemble des variations syntaxiques possibles. Par exemple, étant donnée la phrase *« John loves Mary »*, la famille Tnx0Vnx1[1] de la grammaire XTAG nous permet d'identifier 39 variations syntaxiques possibles dont *« John loves Mary », « Mary is loved by John », « John who loves Mary », « Mary whom John loves », « Mary who is loved by John », « the loving of John by Mary », etc.*

Nous analysons ensuite ces phrases avec le Stanford Parser et définissons pour les graphes de dépendances produits par l'analyseur[2] des règles de réécriture permettant de réécrire chacune des structures produites en une forme normalisée. Par exemple, des règles sont définies pour réécrire chacun des arbres de dépendances donnés en Figure 2 et 3 en une structure unique, la structure donnée en Figure 4. Plus généralement, à partir des analyses correctes produites par le Stanford Parser pour chacune des variantes syntaxiques d'un type de phrase donné (e.g., phrase avec un verbe transitif), nous définissons un ensemble de règles de réécriture permettant

[1] classe des verbes régissant deux arguments nominaux

[2] Comme le Stanford Parser retourne plusieurs analyses pour une même phrase, nous ne considérons bien sûr que les structures correctes pour la définition des règles de réécritures.

de transformer les arbres de dépendances de chacune de ces variantes en une représentation sémantique unique.

Pour évaluer l'impact de notre procédure de normalisation, nous avons défini des règles de réécriture pour la classe des verbes transitifs et celle des verbes intransitifs. Pour ces deux classes, la normalisation des structures requiert la définition de 32 règles de réécriture. Les règles définies posent des contraintes sur les dépendances entre les mots, ainsi que sur la catégorie syntaxique de certains mots. Ce deuxième type de contrainte est nécessaire pour les arbres incluant un verbe utilisé comme adjectif (i.e., « the formated disk »), afin d'éviter de réécrire tous les graphes contenant un adjectif (i.e., « the beautiful girl »). Pour vérifier qu'il s'agit bien d'un emploi adjectival d'un verbe, la contrainte spécifiée demande que le mot étiquetant le noeud considéré appartienne à la catégorie verbe de *WordNet*[3].

Ainsi, les règles représentés graphiquement sur les figures 5 et 6 sont définies de la manière suivante en GrGen.Net :

```
rule nx0Vnx1 {
 pattern{
  verb:element;
  if{verb.verb != "None";}
  np0:element;
  np1:element;
  verb -:nsubj-> np0;
  verb -:dobj-> np1;
 }
 replace {
  verb -:arg0-> np0;
  verb -:arg1-> np1;
 }}
```

```
rule nx1Vbynx0 {
 pattern{
  verb:element;
  if{verb.verb != "None";}
  np1:element;
  be:element;
  np0:element;
  verb -:nsubjpass-> np1;
  verb -:auxpass-> be;
  verb -:agent-> np0;
 }
 replace {
  verb -:arg0-> np0;
  verb -:arg1-> np1;
 }}
```

Pour optimiser leur bonne application, les règles sont exécutées par ordre de spécificité en allant de la plus spécifique à la moins spécifique[4]. Par exemple, la règle de réécriture pour le passif long est appliquée avant celle pour le passif court, assurant ainsi la réécriture du groupe prépositionnel en « by ». En outre, lorsqu'une règle s'applique à plusieurs sous-structures du graphe réécrit, la réécriture s'applique simultanément à l'ensemble de ces sous-structures (application globale plutôt qu'unique).

2.3 Traitement logique de l'implication textuelle

Pour déterminer si un texte T_1 implique textuellement un texte T_2, nous traduisons les arbres de dépendances en formules logiques puis nous testons la validité de l'implication logique en logique du première ordre.

[3] http://wordnet.princeton.edu

[4] Techniquement, la stratégie de réécriture déployée est la suivante : chaque séquence de règles contient une seule règle et l'ensemble des séquences est combiné par concaténation en ordonnant les séquences par ordre de spécificité (une règle R_1 est dite plus spécifique qu'une règle R_2 si l'ensemble des graphes pouvant être filtrés par R_1 est inclus dans l'ensemble des graphes pouvant être filtrés par R_2).

Traitement logique La transformation d'un graphe de dépendances en formule de la logique du premier ordre se fait de la façon suivante. À chaque noeud du graphe correspond une variable quantifiée à laquelle est appliqué un prédicat unaire ayant pour nom le mot associé au noeud. Les arcs introduisent une prédication binaire ayant pour nom l'étiquette de l'arc et pour arguments les variables associées à ses noeuds source et destination. La formule finale est la conjonction des prédications ainsi obtenues. Ainsi pour le graphe de la Figure 2 nous avons la formule $\exists x, y, z : love(x) \wedge john(y) \wedge mary(z) \wedge nsubj(x, y) \wedge dobj(x, z)$.

Implication textuelle La validité de l'implication logique entre deux formules peut être vérifiée grâce à des outils formels comme les prouveurs de théorème ou les constructeurs de modèles. Comme le montre (Blackburn *et al.*, 1999) , ces deux outils peuvent être utilisés de façon complémentaire. En effet, un prouveur de théorèmes vise à dériver une contradiction tandis qu'un constructeur de modèles cherche à créer un modèle satisfaisant la formule d'entrée. En conséquence, un prouveur est souvent peu efficace pour une formule satisfiable tandis qu'à l'inverse, un constructeur de modèles est souvent peu performant pour une formule non satisfiable. Pour tester la relation d'implication $\phi_1 \rightarrow \phi_2$ entre les représentations loqiques de deux textes T_1 et T_2, nous utilisons donc en parallèle le prouveur de théorèmes Equinox[5] et le constructeur de modèles Paradox[7]. Dans les deux cas, la requête est la négation de l'implication $\neg(\phi_1 \rightarrow \phi_2)$ et une réponse positive indique une non-implication textuelle.

En pratique, nous ne testons pas l'implication entre deux représentations mais entre plusieurs. Nous considérons en effet les 5 premières sorties de l'analyseur pour chacune des deux phrases considérées et cherchons à détecter une relation d'implication entre au moins une paire de représentations. Plus précisément, étant donné deux phrases T_1, T_2 et les formules logiques $T_1^1, \ldots, T_1^5$ et $T_2^1, \ldots, T_2^5$ associées à ces phrases, T_2 est une implication textuelle de T_1 si $(T_1^1 \rightarrow T_2^1) \vee (T_1^1 \rightarrow T_2^2) \vee \ldots$ ou par équivalence, $(T_1^1 \wedge \ldots \wedge T_1^5) \rightarrow (T_2^1 \vee \ldots \vee T_2^5)$.

3 Évaluation

Afin d'évaluer notre système de normalisation, nous avons développé une suite de tests illustrant la variation syntaxique entre deux paires de phrases.

À partir d'un lexique restreint et d'expressions régulières décrivant les structures syntaxiques possibles pour un verbe et les représentations sémantiques associées, nous avons développé un script permettant d'engendrer des paires phrases-sémantiques telles que les phrases contiennent soit un (phrase simple) soit deux ou plus (phrase complexe) verbes conjugués. Après avoir manuellement vérifié la correction des éléments générés (paires phrase-sémantique), nous avons utilisé ces paires pour créer les éléments de la suite de tests c'est-à-dire, des paires de phrases annotées avec la valeur VRAI ou FAUX selon qu'il y a ou non une relation d'implication textuelle entre les membres de la paire.

La suite de tests résultante[6] comporte 2520 paires de phrases annotées dont 937 correspondent à une implication entre une phrase (complexe ou simple) et une phrase simple (i.e.,*1V+IT* – e.g., implication (2) de la Figure 1), 437 à une implication entre deux phrases complexes (i.e.,*2V+IT*

[5] http://www.cs.chalmers.se/ koen/folkung/

[6] accessibles à l'adresse suivante : http://www.loria.fr/~bedaride/publications/taln08-bedgar/tests.html

– e.g., implication (1) de la Figure 1) et 1146 à une non-implication (i.e.,*V-IT* – e.g., non-implication (3) de la Figure 1).

Pour chaque élément dans cette suite de tests, nous testons l'implication selon la procédure décrite dans la section précédente : pour chacune des deux phrases dans une paire, les 5 premières analyses du Stanford Parser sont transformées en formules de la logique du premier ordre et il est testé si ou non, la conjonction des 5 représentations de T_1 implique la disjonction des 5 représentations de T_2. Le test est fait d'une part, à partir des graphes de dépendances produits par l'analyseur et d'autre part, à partir des graphes normalisés.

On obtient les résultats suivants :

type de graphe	réponse	*1V+IT*	*2V+IT*	*V-IT*
graphe dépendances basique	+IT	632 (67.4%)	26 (5.9%)	122 (10.6%)
	-IT	290 (30.9%)	406 (92.9%)	1021 (89.1%)
	Échec	15 (1.7%)	5 (1.2%)	3 (0.3%)
graphe dépendances normalisé	+IT	911 (97.2%)	369 (84.4%)	134 (11.7%)
	-IT	12 (1.3%)	58 (13.3%)	1011 (88.2%)
	Échec	14 (1.5%)	10 (2.3%)	1 (0.1%)

La logique du première ordre n'étant que semi-décidable, la ligne « Échec » recense les cas où les raisonneurs échouent à donner une réponse. On observe qu'en pratique le nombre de cas indécidables reste limité.

Pour les implications de phrases simples (première colonne, *1V+IT*), la normalisation des graphes de dépendances permet un gain en précision de 29.8% : sur 937 cas d'implications textuelles, 911 sont reconnues comme telles contre seulement 632 sans normalisation. Le gain augmente de façon significative pour les implications de phrases complexes (*2V+IT*) avec 369 cas reconnus avec la normalisation contre 26 sans, soit un gain en précision de 78.5%. Enfin la dernière colonne (*V-IT*) montre que la normalisation n'est pas trop permissive puisque avec ou sans normalisation, le taux de non-implications reconnues est d'environ 90%.

L'analyse manuelle des résultats de l'approche normalisante montre que les cas d'erreurs sont dûs d'une part, à des cas où les 5 premières sorties de l'analyseur ne contiennent pas le graphe correct pour la phrase analysée et d'autre part, à un petit nombre de règles de réécriture incorrectes.

4 Comparaison avec l'existant

Notre approche présente des similitudes avec l'approche présentée dans (Fabio Massimo Zanzotto & Moschitti, 2007) qui elle aussi utilise des règles de réécriture pour détecter l'implication textuelle. Les différences sont de deux ordres. Premièrement, les règles de (Fabio Massimo Zanzotto & Moschitti, 2007) portent sur les arbres syntaxiques plutôt que sur des graphes de dépendances. Deuxièmement, ces règles sont acquises automatiquement à partir des corpus d'implications textuelles plutôt que manuellement à partir d'une grammaire.

Les graphes de dépendances représentent une abstraction sémantique des arbres syntaxiques, ils permettent donc d'avoir des règles de réécritures plus génériques. Par exemple, le passif

simple est géré avec une seule règle avec les graphes de dépendances alors qu'avec les arbres syntaxiques dont la structure varie significativement avec le temps du verbe, il faut avoir une règle pour chaque temps.

La génération de règles de réécriture à partir de paires de textes impliqués à l'avantage d'être totalement automatisée, mais les règles obtenues prennent chacune en compte un ensemble de variations syntaxiques et sémantique, ce qui les rend peu modulaires et très spécifiques : un règle ne pourra être appliquée que si les phrases analysées présentent l'ensemble des caractéristiques syntaxiques et sémantiques décrites par le motif de filtrage de cette règle. Par contraste, l'approche que nous présentons permet la réécriture du noyau verbal indépendamment des contextes spécifiques dans lequel il apparaît. La réécriture sémantique (de structures synonymiques par exemple) n'est pas traitée pour l'instant mais pourrait l'être dans une deuxième système de réécriture qui seraient combiné, de façon modulaire, avec le système de normalisation syntaxique présenté ici.

5 Conclusion

Les évaluations présentées dans cet article montrent d'une part, que les analyseurs existants produisent des structures distinctes pour des phrases ayant le même contenu informationnel et d'autre part, qu'il est possible d'utiliser les systèmes de réécriture pour normaliser ces structures et leur associer une représentation sémantique unique.

Pour les exemples considérés, la normalisation permet un gain en précision conséquent (variant entre environ 30 et 80% selon la complexité syntaxique des phrases considérées). La méthode proposée a donc un potentiel clair pour améliorer la capacité des systèmes de traitement d'implications textuelles à traiter de la variation syntaxique.

Pour passer à échelle et obtenir un système de détection d'implications textuelles qui, à partir de cette méthode, traite de l'ensemble des variations syntaxiques, la méthode doit être généralisée à l'ensemble des cadres syntaxiques et des alternations. Nous travaillons actuellement au développement d'un système de « méta-règles » qui permet de dériver, à partir des règles de réécriture définies pour les verbes transitifs et intransitifs, des règles de réécriture pour les autres classes de verbes (par exemple, verbes à objet prépositionnel, verbes à objet phrastiques, verbes ditransitifs) ainsi que pour les alternances.

Comme nous le mentionnons dans l'introduction, la variation syntaxique, si elle est un important facteur de la reconnaissance d'implications textuelles, n'en n'est pas pour autant le facteur unique. Il est également nécessaire de pouvoir prendre en compte de nombreux autres facteurs de variations dont en particulier, la sémantique lexicale (utilisation de synonymes, hyperonymes, variantes terminologiques, etc.). A moyen terme, nous envisageons d'enrichir le système de règles de réécriture pour prendre en compte les connaissances lexicales contenues dans WordNet, VerbNet et/ou FrameNet.

Références

BLACKBURN P., BOS J., KOHLHASE M. & DE NIVELLE H. (1999). Inference and computational semantics. In H. .BUNT & E.THIJSSE, Eds., *Proceedings of the Third International*

Workshop on Computational Semantics (IWCS-3), p. 5–19.

DAGAN I., GLICKMAN O. & MAGNINI B. (2005). The PASCAL Recognising Textual Entailment Challenge. In *Machine Learning Challenges Workshop*, p. 177–190.

DE MARNEFFE M.-C., MACCARTNEY B. & MANNING C. D. (2006). Generating typed dependency parses from phrase structure parses. In *In Proceedings of LREC-06*.

FABIO MASSIMO ZANZOTTO M. P. & MOSCHITTI A. (2007). Shallow semantics in fast textual entailment rule learners. In *In Proceedings of the Third Recognising Textual Entailment Challenge*, p. 72–77.

GROUP X. R. (2001). *A Lexicalized Tree Adjoining Grammar for English*. Rapport interne IRCS-01-03, IRCS, University of Pennsylvania.

KLEIN D. & MANNING C. (2003). Accurate unlexicalized parsing.

KROLL M. & GEIŠ R. (2007). Developing graph transformations with grgen.net. In *Applications of Graph Transformation with Industrial releVancE - AGTIVE 2007*. preliminary version, submitted to AGTIVE 2007.